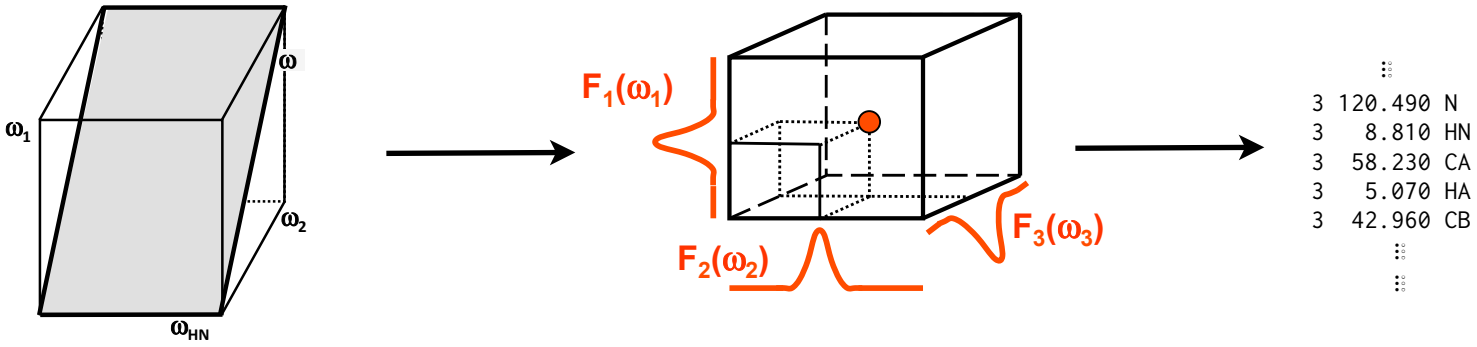
UNIVERSITY OF GOTHENBURG



deco - A decomposition program for NMR projection experiments

Master of Science Thesis in the computer science program

JONAS FREDRIKSSON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, January 2012

deco - a decomposition program for NMR projection experiments

Jonas Fredriksson

Examiner: ARNE DALHBERG

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: From fast projection experiment an assignment of a protein can be done with the right software.

Department of Computer Science and Engineering
Göteborg, Sweden May 2011

Abstract

In protein NMR high dimensional spectra are difficult to measure because long measurement times and low signal to noise. To overcome this barrier fast acquisition techniques have been developed. One of these is projection experiments that uses coupled evolution periods instead of using independent evolution periods. This reduces measurement time dramatically but has constraints in terms of protein size because of low signal to noise present in high dimensional spectra. Analyzing these spectra requires software tools that can process the convoluted spectra and assign the resulting signals. In this thesis a new algorithm is presented for decomposing convoluted spectra using python as a model and Fortran routines for steps that are time critical. All test where done on the azurin and the ubiquitin protein. and the result is a software tool for resolving projection experiments using a new algorithm together with a graphical user interface.

Table of contents

Introduction

NMR is a versatile method for solving structural and dynamic problems related to proteins, peptides and smaller molecules[1] [2] and it has become increasingly important since the number of expressed proteins have increased[3]. The technique is based on spin properties in atoms and their interaction with a strong external magnetic field. In the strong external field all spins are distributed according to Boltzmann statistics. A radio frequency pulse can then tilt all the spins in a plane perpendicular to the external magnetic field. Depending on the molecular environment the spins induces a different electromagnetic field when precessing in the perpendicular field. This analog signal can then be Fourier transformed giving every signal a corresponding frequency that can be evaluated in different ways. The different frequencies correspond to different parts of the molecule and contains information about the nuclei and the surroundings. Assignment in the frequency is to map every signal to every nucleus in every residue. This can then be used for analysis of the frequency shapes, relaxation studies and structure evaluation[4].

Protein NMR requires usually the recording of more than two dimensions to resolve all frequencies[5] [6]. The adding of dimensions increases measurement time exponentially putting a limit to the number of dimensions that can be recorded in an experiment. Because of this, alternative techniques have to be developed to shorten measurement time while still trying to obtain the same information as in regular NMR protein experiments[7] [8]. One of these are projection decomposition techniques that record experiments in a fast time frame with coupled evolutions periods that gives 2D projection experiments from higher dimensional experiments[9]. A significant time saving is achieved using this technique. Fourier transformation of the time domain signals results in linear combinations of frequencies. Now the indirect frequency axis contains the sum or the difference between the two or more nucleus instead of a single frequency for every nuclei. Figure 1 shows a 45 degree projection in frequency domain.

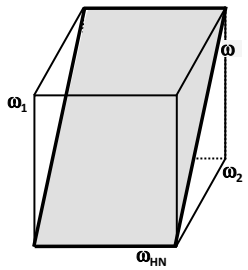


Figure 1. 45 degree projection in a 3D array of data points in frequency domain.

Here Ω now contains information for both Ω_1 and Ω_2. If the whole cub was going to be measured with 100 complex points in the Ω_1, Ω_2 dimension this experiment would take 13 h. A projection with 0, 90, 45 and -45 projection angles would take 13 min with the same number of complex points. This time saving would increase even more for 4D and 5D experiments. Thus projection decomposition techniques are feasible for reducing experimental time in protein NMR spectroscopy[10].

Projection decomposition

Regular NMR experiments can be described as a summation of time domain signals[11] over N dimension experiments:

(1) $$S(t_1,t_2 \ldots t_N) = \sum_k f_1^k(t_1) \otimes f_2^k(t_2) \ldots \otimes f_N^k(t_N)$$

Equation 1 describes how time domain signals can be described as a direct product between functions describing the time domain signal. All signals are enumerated by k. Every set of k:s are described as components. Every component have N number of shapes that are enumerated from 1 to N in formula 2. All valid signals in a spectra are then described by all k. In the case of azurin that would correspond to 128. These can be seen in figure 6. Prolines are not shown because lack of NH. Fourier transform over all signals in equation 2 gives the corresponding equation in frequency domain:

(2) $$S(\omega_1,\omega_2 \ldots \omega_N) = \sum_k F_1^k(\omega_1) \otimes F_2^k(\omega_2) \ldots \otimes F_N^k(\omega_N)$$

This equation describes the corresponding frequencies after Fourier transform of equation 1. In multiway decomposition time delays in the mixing time in pules are coupled:

(3) $$e^{(R_1 + i\Omega_1)t} \times e^{(R_2 + i\Omega_2)t}$$

Here R are the two relaxation parameters, Ω frequencies of the signal and t the coupled evolution periods. This essentially means that time domain signals now are coupled for nucleus in the indirect dimension. With these depending coupling scheme Equation 1 has the following form.

(4) $$p_m(t,t_{\mathrm{N}}) = \sum_k (f_1^k \cdot f_2^k \ldots)(t) \otimes f_N^k(t_{\mathrm{N}})$$

Now the indirect signals are multiplied before the direct dimension. Direct multiplication is now done against the direct dimension which is described by the f_N function where N is the dimensionality of the experiment. Fourier transform of equation 4 gives the following expression:

(5) $$P_m(\omega,\omega_{\mathrm{N}}) = \sum_k (F_1^k * F_2^k \ldots)(\omega) \otimes F_N^k(\omega_{\mathrm{N}})$$

Equation 5 describes how the frequency signals can be described as a convolution between the frequencies in the indirect dimension[12]. The convolution of two functions that are Fourier transformed describes an addition or subtraction of the frequencies. This addition of frequencies in the indirect dimension creates an turn around of the signal. In equation 5 we have a recorded spectra in the right side and a model describing this spectra on the left side. A minimization of these two entities forms the basis of projections decomposition:

(6) $$\min(\| P_m(\omega,\omega_N) - \sum_k (F_1^k * F_2^k \ldots)(\omega) \otimes F_N^k(\omega_N) \|^2)$$

In equation 6 noise are not considered. This can sometimes be added by using a Tikhonov regulation factor in the equation[13]. Discrete convolution between two vectors can be described as:

(7) $$(a*b)_i = \sum_j a_j \cdot b_{i-j} \qquad i.j = -n \text{ to } n$$

Equation 7 describes that a ”wraparound” effect will take place when i-j is negative. This correspond to a convolution of two frequencies that are outside the spectral sweep width. Convolution for 45 degree projections can equivalently be written in the following way:

(8) $$M = \begin{pmatrix} a_0 & a_{N-1} & a_{N-2} & \cdots & a_1 \\ a_1 & a_0 & a_{N-1} & \cdots & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_3 \\ \vdots & \vdots & \vdots & & \vdots \\ a_{N-1} & a_{N-2} & a_{N-3} & \cdots & a_0 \end{pmatrix}$$

The matrix for describing -45 degrees convolution is done by swapping the columns around the central column. The matrix vector product Mb is the convolution between two vectors where one is described in a matrix form M. This forms the basis for a convolution equation

(9) $$Mb = P$$

where P is convolution between the two vectors. This equation can be used for decomposition calculations by using P as the measured spectra and b for the unknown frequency. The M matrix is the known frequency expressed in equation 8. By using this it is possible to calculate the unknown frequency. Equation system 9 is solved by using nnls, a non negative least square solver. The residual is used for estimating the difference between Mb and P. Equation 9 where used in all optimization steps.

Previous implementations

The first implementations of the decomposition algorithm were called prodecomp and used Matlab[14] as the language for the implementation. This proved to be slow for larger input of data and also could not be used for decomposition of larger spectra. Also no graphical interface existed thus making it hard to use for non experts. The two following implementations used python with numpy and scipy libraries[15] for numerical calculations. These versions was faster than the first matlab implementation and a comparison can be seen figure 1 between the Matlab version and the two python versions[16]:

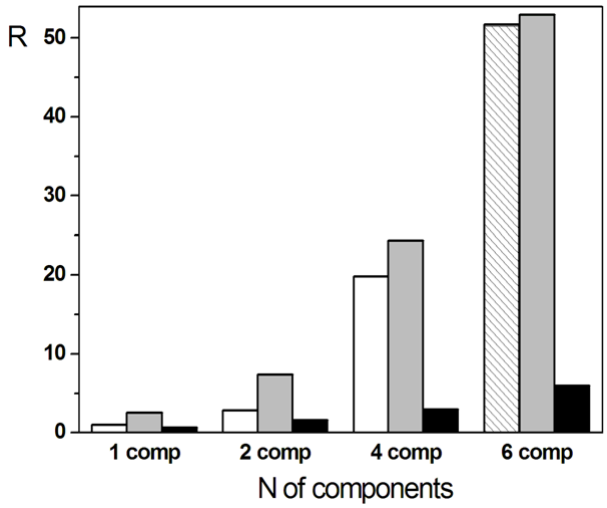


Figure 2. Comparison between the three versions of projection implementations.

In figure 2 three versions of prodecomp are compared with regard to running times and number of components. The white and the grey boxes indicate matlab and an early python version. The black box was the latest version with improved matrix handling and less memory consumption witch made it possible to run larger datasets. However the python version was still slow when larger amount of data was used and the whole spectra could not be analyzed at the same time. To overcome this the spectra had to be divided into smaller parts for the analysis. No external functional calls was made in the two python version and the nnls function was translated verbatim from the matlab version. An early C version was developed but discarded at an early stage because complex memory problem when handling matrices. As an example of the memory constraints that occur when calculating a full spectra once instead of divide it into smaller parts, consider two projection experiments covering the backbone of a protein. Total number of planes would be 38. Assume that every plane would have 800*128 data points and that a double precision would be used. Using all planes at once with no slicing would give a matrix of size of 35 Mb. These arrays are then used in a kronecker product operation where the resulting plane would be an 102400x128 sized array and this would slow down the computation. By using projection experiments with less dimensionality this problem can be avoided because of only three planes that are used instead of 38 planes.

Running times

The first step in the development was to profile the previous version of the decomposition program. This was done using the cProfile module in python. The data was from two backbone experiments on Azurin using all 38 planes with 800 points in the direct dimension and 128 points in the indirect dimension. The interval was 742 to 744 points and two components where used. A selection of the results are shown in table 1:

tottime	cumtime	function
0.455	11.894	prodecomp
2.168	5.364	midmatrix
0.558	3.193	fastnnls
0.971	3.102	roll
1.572	1.586	dot
0.980	0.982	concatenate

Table 1. Selection of profiler results showing the functions that takes most time in the old decomposition method.

The function roll changes array elements along a given axis and midmatrix loops over matrices, all used for the creation of the convolution matrix explained further down. Looping and matrix multiplication is a bottleneck as so is also fastnnls. To further investigate different options a comparison was made between python, C and Fortran to compare running times between these with regard to matrix multiplication, looping and nnls in order to replace functions to improve performance. As a first step, matrix multiplication and looping over matrix was compared between python, Fortran and C. Randomized square matrices was used with sizes from 100 to 5000 in 500 step intervals. These where used for matrix multiplication with floating numbers and setting numbers over loops. The result is shown in figure 3:

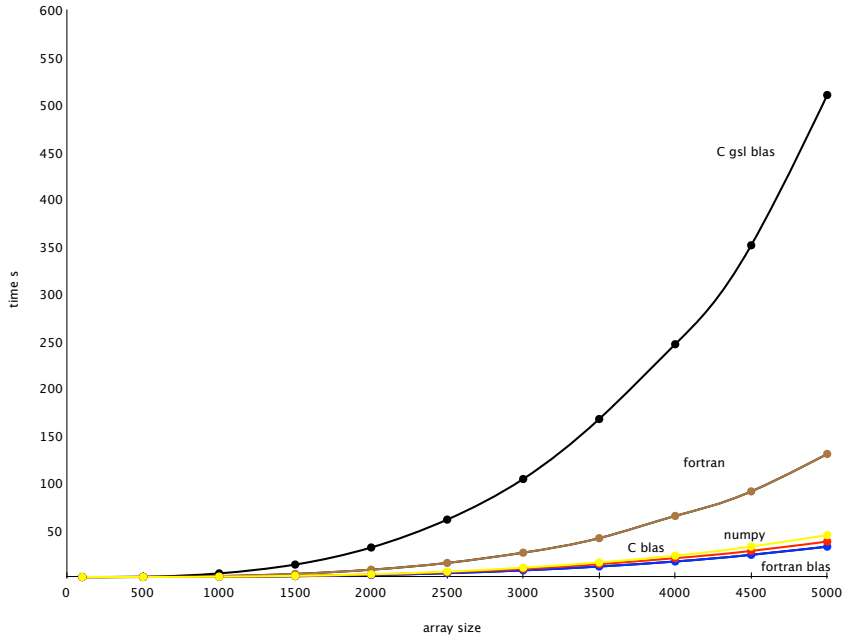


Figure 3. Different runnings times for matrix multiplication.

Here it's clear that gsl is performing worse than Fortran and numpy. numpy is using Atlas library which is an optimized library for numerical calculations. gsl uses gslblas wich is slower than the netlib blas implementation. Matrix multiplication is one of the bottlenecks in the present implementation of the decomposition software. All three languages that use Fortran blas libraries are performing equally well with an slight edge for Fortran. The conclusions is that numpy matrix multiplication is fast but calling a Fortran routine will add a slightly speed gain.

Another bottleneck is looping over different elements in a matrix. A comparison with Fortran and C showed clearly that python is slower than both these languages. Here a empty n*n matrix was used to fill with integers. The different sizes where 100,500,1000...5000. The result is shown in figure 4:

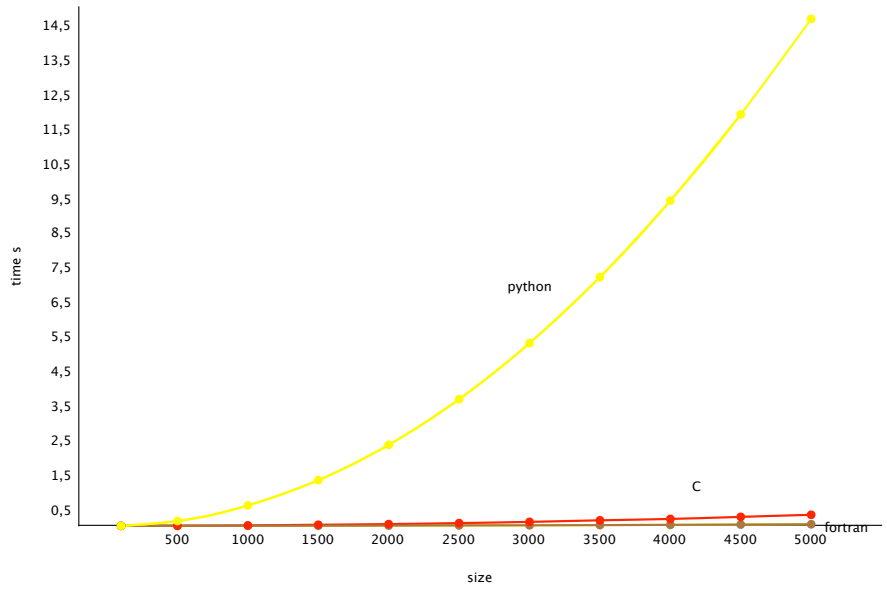


Figure 4. Different runnings times for looping.

Fortran is the fastest of the three, mostly because it uses parallel assignment if possible in the forall loop that fills a matrix without any specific order. Even though the different between C and Fortran was small in absolute numbers, when using several loops in the implementation this difference will come to play. The current version uses an in house implemented version of fastnnls that was based on the matlab implementation. This version is compared to nnls from scipy and nnls from Fortran with input matrixes as before, randomized square matrixes where used in different sizes. The sizes used where the same as in the previous example. The result is shown in figure 5.

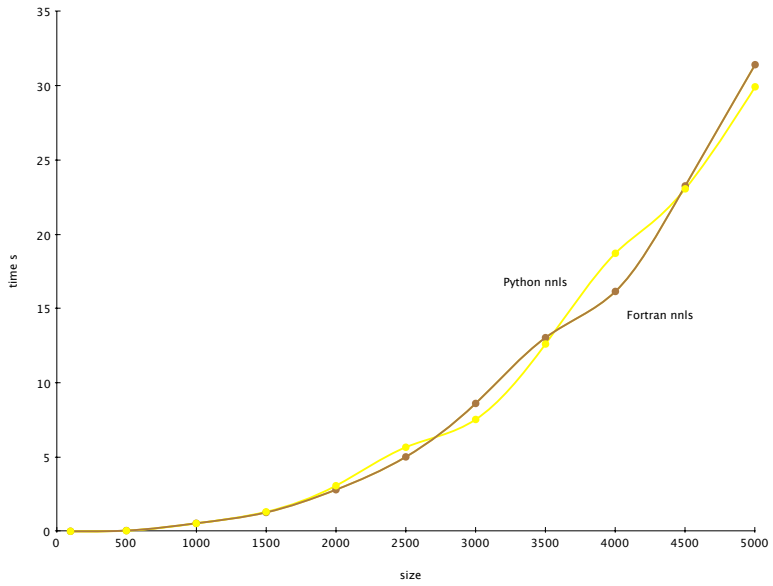


Figure 5. Different running times for nnls implemented in Fortran and python.

Here it is clear that the numpy version that uses a Fortran 77 routine did not differ from the Fortran 90 implementation significantly. Existing version for C did not exist so C was excluded from the comparison. A small difference was seen in the range 3500 to 4500, this could probably be due to external factors, or an uneven spread of the random numbers generated. Another test was to test the main decomposition program against different implementations that used C and Fortran routines to compare different running times. Here it was seen that C calls via ctypes[17] interface improved execution speed to a certain degree but also created an unnecessary overhead with regard to arguments passing between the two languages. As described below, using linked Fortran object files provided to be a much easier solution while still having a significant speed gain compared to python functions. Apart from using Fortran routines in the python code instead of c functions a reduction of the number of input parameters was necessary to decrease unnecessary recalculations. So the initial approach was to use one point instead of several points and one component. This was tested but it turned out that an increased interval would be more beneficial with a one point assumption in every interval.

Implementation

The existing decomposition algorithm suffer from a few drawbacks as mentioned above. Three parameters have to be provided to make the decomposition which creates many possible combinations witch can introduce erroneous components, i.e crowded spectra with many signals that overlap may alter the final result. Another problem is when strong signals are present close to weak signals. These strong signals are then often present in the components of the weaker components, influencing the resulting shapes making it hard to distinguish the right shape from the wrong one. A third problem is a lack of a comparing criteria for the resulting shapes. All these problems have to be addressed in a modified decomposition method and one possible path is to reduce the interval to one point thereby reducing the number of parameters to one. The projection experiments used in the previous implementation used all possible combinations from the projection experiments in the decomposition procedure. This approach created many planes when the number of dimensions increased to 4 and 5 as described in equation 10:

(10)
$$p = \sum_{x=1}^{n} \frac{n!}{(n-x)!x!} \cdot 2^{x-1}$$

Equation 10. The number of planes for a N dimensional projection experiment where n is the number of indirect dimensions.

This approach was somewhat useful for spectra with low signal to noise but it also extended the measurement time considerably. For example, the number of planes for a 5D (n=4) experiment are 40, with the last 8 planes having the lowest signal to noise and at the same time taking almost half of the total experimental time due to the recording of all 8 combinations. Low signal to noise could in fact reduce the accuracy of the calculations by introducing more noise than signals. By reducing the dimensionality of the spectra from 5D and 4D to 3D the measurement time decreases and individual variations can be done with different parameters depending on the nuclei measured. With this setup, more experiments are required to cover the same set of nuclei recorded for a higher dimensional projection experiment and only combinations of two nuclei are used in the algorithm but the signal to noise can improve and individual experiments can be adjusted with respect to number of scans and indirect points.

Overall Structure

The programs used for testing and development of the decomposition algorithm can be divided into two parts: python testing programs and final python programs with additional Fortran subroutines. Python was used because it supports numerical calculations and prototyping and testing can be done very fast compared to other languages and also be able to call other languages. The python programs consists of helping programs and develop programs. Helping programs are used for different tasks like displaying a spectra or convert points to ppm. The develop programs are testing programs used for different calculations.

The main programs that was used to try different decompositions algorithms on a subset of the Azurin protein where the following: finalDeco.py, finalDeco.f95, decofunctions.f95, conversion.py, readFT2files.py, plotProjections.py. The graphical interface consists of the gui3.py program and here ubiquitin was used as test data. Testing data used for developing the deco algorithm came from projection experiments made on the protein Azurin[18]. Two different experiments where used that covered the backbone nuclei for two connected residues, thus giving information about the current and the previous residue. All 38 planes where recorded but only planes containing the N nuclei and a combination of one the other nuclei or a single nuclei, X={CO,Cα,Cβ,Hα,Hβ,Cαβ,Hαβ} where used as an input for all programs in order to test 3D to 2D backbone projections experiments. In total three planes where used: (N+X, N-X, X). The single N plane was used in the final selection. For every test a peak representing a residue from different parts of the corresponding ^{15}N-HSQC plane where used. Every plane was displayed using the program plotProjections.py and in figure 6 is the ^{15}N-HSQC plane from Azurin used for peak picking.

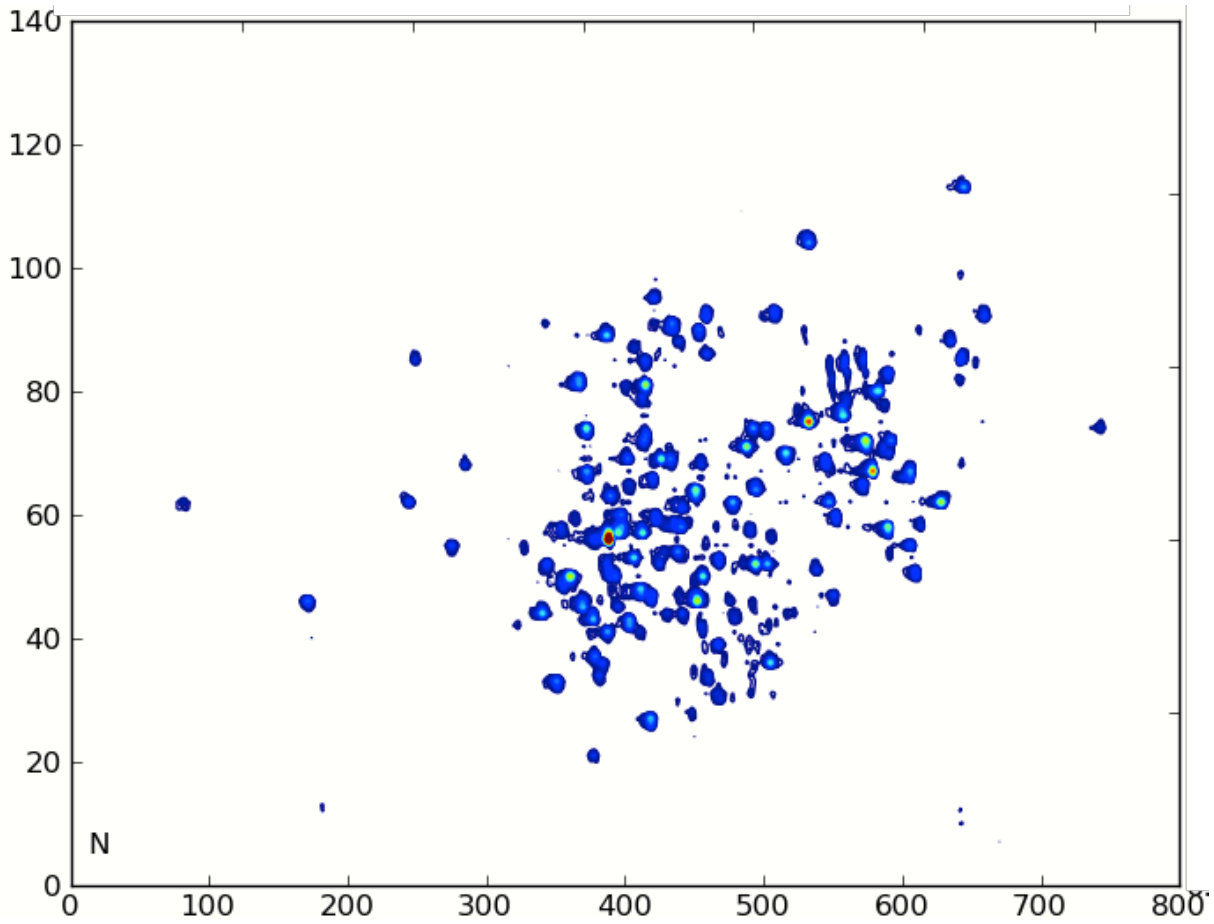


Figure 6. Corresponding ^{15}N-HSQC projection spectra from azurin used to select different residues. Note that side chains and prolines are not present in the projection ^{15}N-HSQC. Units on x and y axis are in points.

In figure 6 units on the x axis corresponds to direct points while units on the y axis corresponds to indirect points. As can be seen, intensity differs among the peaks. In a regular ^{15}N-HSQC spectra these weak peaks would have a much higher intensity due to less magnetization transfer steps while in projection experiments signal to noise is reduced which influence decomposition calculations negatively. All development and testings where used on the following residues from Azurin.

interval x (pts)	interval y (pts)	residue	comment
171	46	46	strong single peak
397	60	34	weak, close to
395	57	103	strong residue
377	21	81	strong overlapped peak
343	91	88	isolated glycine
182	13	114	weak isolated
81	62	38	weaker left most residue
421	95	9	glycine residue
419	27	128	n-terminal
606	55	30	has a close peak to 30
606	67	73	same peak as 55?
601	66	65	same interval as 30
428	54	18	residue in the middle
388	56	84 or 2?	strongest peak in spectra

Table 2. Azurin residues used for testing the algorithm. The last residue is ambiguous.

These residues represents different situations (low signal to noise, overlap of other peaks, crowded regions) that an algorithm have to be tested against. The testing algorithm used on these intervals uses only one fixed interval and one component, one for every actual residue. The rest of the signals can in this case be either signals from neighboring residues or noise which is equivalent from the algorithm standpoint. The spectra was recorded using two backbone projection experiments covering the current and the previous nuclei. The last residue had the strongest signal but lacked a clear reference. This can in fact be an artifact of the experiment. The closest reference was residue 84 and 2. The residues where chosen to be an representative subset of the total number of residues which where 123 and 5 prolines not present in the spectra giving a total of 128 residues.

Development

The first assumption was to use peak picked N and NH shifts for all projections that contained the N nucleus. This assumes that the peak picker can correctly peak pick all valid residues neglecting side chains that are present in a regular ^{15}N-HSQC and that the shift variations is very small between the N shifts of all the involved planes. There exists different peak pickers for NMR spectra. The one used here was nmrDraw which is a part of the nmrPipe software package[19]. This peak picker uses different symmetry criteria to determine peaks from noise and threshold levels can be adjusted to decrease or increase number of valid peaks. All valid peaks are identified by their (x,y) coordinates in the direct and the indirect dimension. By using a fixed peak an assumption is made that the picked peak is considered valid for all subsequent calculations, NH peak is the same for all planes. For every signal a vector N is initialized to zero except the peak picked N point that is set to one. For every peak picked point a set of points is used, i.e if the selected point is 171 (residue 46) in the direct dimension then a range from 169 to 173 points in direct dimension are used for the calculation. The slice defined have to be reasonably small otherwise overlapping strong neighboring signals will disturb the calculation. A fixed N also increases the correctness of the calculation because it doesn't have to be optimized. This vector is then used for generating the M matrix by using equation 8. The matrix for calculating negative projection angles (-45 degrees) is then done by shifting all columns in matrix M around the mid vertical column. This matrix is here called Mn. By setting the known signal to one and the rest of the signals in the vector to zero gives an M matrix that merely shifts the convolution spectra (N±X) to the right position. it also enhances the signal. When constructing the M matrix all points has to be shifted half of the indirect dimension points, i.e 64 points for a spectra with 128 points in indirect dimension. This "wrap around" effect takes care of the folding of the spectra, an effect caused when a larger sweep width is used than the final sweep width in the indirect dimension. When developing the algorithm, a first approach was to construct another vector for all signals except the ones for the specific residue. If the signal vector is called f_s and the rest of the signals f_n then slice $c_l=f_s+f_n$ would be composed of the right signal and the rest of the signals would be considered to be noise or other signals. This noise plus the rest signals vector f_n would then be used to subtract all signals except the selected from the spectra, i.e one calculation for the actual signal and one for the rest of the signals. Further test concluded that this was not a useful strategy because only a limited number of points where shifted. Also the projection N+X and N-X are used to construct convolution matrices that are used in the calculation for X presented in equation 10 below. All four matrices are used to determine b in equation 9 by using the nnls algorithm. Note that the corresponding signal in b, the unknown in equation 9 corresponding to the unknown signal X in the projections, have the same position when using either M or Mn, i.e the 45 and -45 projection. But the rest of the signals are not always the same. Signals from other peaks have different positions than the actual signal as long as they are not to close to the current signal. This increases the chance to identify the current signal from the rest of the signals. In equation 5, a set components are used to describe the projection spectra. By using just one component this is can thought of that one component describes the signal and the rest describes the noise or signals from neighboring peaks. The use of one point in for the indirect dimension requires well picked peak, otherwise the calculation can easily be misleading and give erroneous results.

Modified algorithm deco

The flowchart describing the overall procedure is presented below. The only parameters used are the width of the direct dimension witch is set to a default of 5 points. The whole ^{15}N-HSQC is divided into slices which correspond to the number of expected residues. Thus, every residue has a x coordinate and a y coordinate corresponding to the direct dimension and the indirect dimension respectively. The direct dimension usually correspond to the NH nuclei while the indirect dimension can be as single nuclei or a combination of both nuclei N and X where X is one of the other nuclei in the experiment, for example X=CO in the N±CO projection experiment. The combination can be either a 45 degree projection (N+X) or a -45 projection (N-X). This approach reduces the former parameters that required an interval definition and also the number of components. The former algorithm used an alternative least square approach[20 21] which required the above parameters and also a Tikhonov regularization factor. Tikhonov regularization factors are used when there is a large fraction of missing data but it can also create distortion in data[22]. At present deco uses no regularization factor since signal intensity from the N signal is changed initially by setting an N vector with a signal of intensity one and the rest is set to zero. The overall flowchart is described in figure 8:

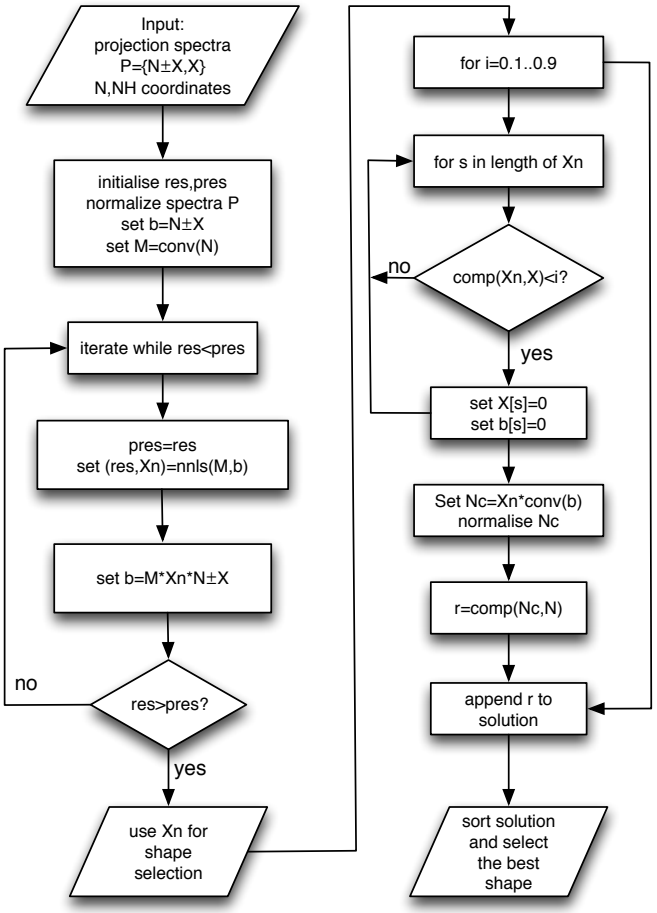


Figure 8. Flowchart describing the general algorithm for decomposition. conv is a function that returns the convolution matrix for an input vector and comp compares two vectors.

The first step in the deco algorithm is to normalize all spectra and to remove negative peaks. The removal of negative peaks is done because the nnls algorithm cannot handle negative input values. Normalization is used to make it easier to handle intensities that has a large spread in intensities and may reduce the importance of the missing Tikhonov regularization factor. The current and the previous residual values are also initialized. A default value of 5 points is used as a preliminary initialization for the width of the direct dimension. The interval defined can be changed depending on the type of protein used, i.e large proteins with a lot of overlapping peaks may require a smaller interval. The convolution matrices M and Mn are determined from the N peak. The iteration step is terminated when the current residual is higher than the previous residual. In the iteration a new b vector is calculated using the calculated Xn vector and the original N±X vector. This is used as an input for the nnls and a new Xn and a residual is calculated as long as the residual is less than the previous calculated. A final selection procedure is then done to select the most probable peak. The details of the iteration is described below in figure 9.

// input
projections={N+X,N-X,X}, peaks coordinates {x,y}

for every N at position x,y:

// setup parameters
set N vector to zero and set N[y] to one
set M,Mn=convolution matrix for vector[N]
initialize x1,x2={x-2 to x+2}

for xi=x1 to x2:

// initialization:
1. normalize projections[N+X] and projections[N-X] at position xi
2. set vectors temp1,b1=projection[N+X] and temp2,b2=projection[N-X]
3. initialize res=99 and pres=100

// optimization
while res<pres

1. set pres=res
2. copy projection[N+X]=b1 and projection[N-X]=b2
3. set Gn,T=convolution matrix for: projection[N+X] and projection[N-X]
4. set Mtot={Gn,T,M,Mn}
5. set b={N,N,projection[N+X],projection[N-X]}
// using equation 10 as an input for nnlsf
6. set Xn[xi],res=nnlsf(Mtot,b)
7. set b1=dot(M,Xn)∗temp1 and b2=dot(M,Xn)∗temp2

```
// selection
initialize final as list

for i=0.1 to 0.9
    for s=1 to length of vector Xn
    // using equation 11,12 in the comparison
        1. set c=comp(Xn,X[x1 to x2]) in the direct dimension
        if c< i
            set Xn[:,s]=0 and X[:,s]=0

    // using equations 13 for final comparison
    2. set bb=dot(transpose(Xn),N[x1 to x2])
    3. set Nc=convolution matrix for b1*bb + b2*bb
    4. normalize Nc
    5. comp(Nc,N) and append score to final list

6. sort final
7. select entry with highest score from final
```

Figure 9. Different parts of the algorithm

The setup parameter is the y determined peak position from the ^{15}N-HSQC. The y index is set to one in a vector initialized to zeros. This vector is then used to construct the M and Mn convolution vector used in equation 9. The range for the interval is also determined, default is 5 points i.e x-2 to x+2 points. Then for every xi in the interval all three spectra X,N+X and N-X are normalized and all negative values are set to zero. Copies of N+X and N-X are made because in the optimization step they will be changed for every iteration but the also the original ones are used in some steps to multiply the newly calculated. Finally current residual res and the previous residual pres are initialized. This concludes the initialization step. The optimization step is an iteration that terminates when the current residual is higher than the current one. For every iteration X is calculated using nnlsf with the current projections N+X and N-X. The resulting calculated X is then used to calculate a new N+X and N-X projection using equations 8 and 9. These new projections are then multiplied by the original ones to get rid of unwanted peaks and noise. A new nnls is now calculated and if the residual is lover than the previous calculated a new set of N+X and N-X projections are calculated that replace the former projections. The loop terminates when the current residual is higher than previous calculated. This is then repeated for every slice of the interval. A final step is to compare every shape in the resulting matrix with direct dimension shape of N. The matrix input for nnlsf is presented below in equation 10. All matrixes on the left side are the Mtot and the right side are the Ntot in figure 9.

(10)

$$\begin{pmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & M(N+X) & \vdots \\ M_{n1} & \cdots & M_{nn} \end{pmatrix}\begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}=\begin{pmatrix} N_1 \\ \vdots \\ N_n \end{pmatrix}$$

$$\begin{pmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & M(N-X) & \vdots \\ M_{n1} & \cdots & M_{nn} \end{pmatrix}\begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}=\begin{pmatrix} N_1 \\ \vdots \\ N_n \end{pmatrix}$$

$$\begin{pmatrix} M_{11} & .... & M_{1n} \\ \vdots & M(N) & \vdots \\ M_{n1} & \cdots & M_{nn} \end{pmatrix}\begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}=\begin{pmatrix} N_1 \\ \vdots \\ N_n \end{pmatrix}$$

$$\begin{pmatrix} M_{11} & .... & M_{1n} \\ \vdots & Mn(N) & \vdots \\ M_{n1} & \cdots & M_{nn} \end{pmatrix}\begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix}=\begin{pmatrix} N_1 \\ \vdots \\ N_n \end{pmatrix}$$

Equation 10. Matrices describing the input for the nnls calculation.

As seen in equation 10, there are four NxN sized matrices that contains the two combined projections Gn $_{(N+X)}$ and T $_{(N-X)}$ and M and Mn. The first two convolution matrices contains the convoluted spectra with the combinations N+X and N-X while the last two convolution matrices contains the N vector used for calculating the X initialized from the HSQC experiment. All NxN convolution matrices are set up according to equation 8. The size of the unknown vector in is the size of the indirect dimension of the processed spectra and it is normally an exponent of two. For example, with an indirect dimension of 128 points the right side matrices becomes 512x128 and the left side 512x1. The unknown vector X is then of size 128x1. This unknown vector form the unknown shape in the convolution equation. The known shapes on the right side of equation 10 are the two N shapes and the two projection shapes. All this is then used in this second step for the nnls algorithm to calculate the unknown shape X out of several possible shapes. The result is X and a residual for the optimization process. The result is the unknown shape calculated using the X,N+X and N-X spectra and the N vector. This is repeated for every x in the range and all calculated X are collected. At this stage the final matrix has the shape 5x128 containing a solution for every interval in the five points range. This matrix is now used to select the final shape with by using cos similarity as in equation 11.

(11)
$$\frac{V_1 \cdot V_2}{\|V_1\|\|V_2\|}$$

Equation 11. Cos similarity between two vectors. The numerator contains the dot product between vector v1 and v2 and the denominator are the norm of the two vectors.

The selection is done by first multiplying the calculated Xn shape with the original X plane to get a final shape in the direct dimension as seen in equation 12. This shape is now compared to all rows of the X plane using cos similarity.

(12)
$$comp: \begin{pmatrix} Xn_{11} & \cdots & Xn_{1n} \\ \vdots & & \vdots \\ Xn_{m1} & \cdots & Xn_{mn} \end{pmatrix} * \begin{pmatrix} Xorg_1 \\ \vdots \\ \\ \\ \\ \\ \vdots \\ Xorg_n \end{pmatrix}, \begin{pmatrix} X_{11} & \cdots s(i) \cdots & X_{1n} \\ \vdots & & \vdots \\ X_{m1} & \cdots s(i) \cdots & X_{mn} \end{pmatrix}$$

Equation 12. Comparing every row on the Xn matrix with the calculated row. S(i) represents every column in the Xn matrix

Cos similarity gives a value between -1 and 1 where -1 and 1 are perfect match and 0 is no match. by comparing all shapes in the original X plane a qualitative value gives information how good the decomposition has been. A set of values are used in increasing order from 0.1 to 0.9 and for every shape that have a lesser value than the current is then set to zero. This is a way of removing shapes that are not valid. A final matrix is then calculated with matrixes in equitation 13 that represent the Nc shape. This Nc is then compared to the original N that was constructed using the peak picker earlier. Finally the selected value that has the highest value is chosen as the final shape.

$$
\begin{aligned}
bb: &\begin{pmatrix} Xn_{11} & \cdots & Xn_{1m} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ Xn_{n1} & \cdots & Xn_{mn} \end{pmatrix} * \begin{pmatrix} x_{11} \\ \vdots \\ x_{m1} \end{pmatrix}, fN2: \begin{pmatrix} N_1 \\ \vdots \\ \vdots \\ \vdots \\ N_n \end{pmatrix} \\
Nc: &\begin{pmatrix} b1_{11} & \cdots\cdots & b1_{1m} \\ \vdots & & \vdots \\ \vdots & M & \vdots \\ \vdots & & \vdots \\ b1_{n1} & \cdots\cdots & b1_{mn} \end{pmatrix} * bb + \begin{pmatrix} b2_{11} & \cdots\cdots & b2_{1m} \\ \vdots & & \vdots \\ \vdots & Mn & \vdots \\ \vdots & & \vdots \\ b2_{n1} & \cdots\cdots & b2_{mn} \end{pmatrix} * bb \\
comp: &\begin{pmatrix} Ncn_1 & \cdots\cdots\cdots & Ncn_n \end{pmatrix}, \begin{pmatrix} fN2_{11} & \cdots\cdots\cdots & fN2_{1n} \end{pmatrix}
\end{aligned}
\tag{13}
$$

Equation 13. The three matrices used for the final comparison. Xn is compared to fN2 and the best score is used to pick the final shaped

Programs

finalDeco.py: this program contains the main algorithm for the decomposition plus additional programs for testing data. The implementation is in python with special calls to Fortran routines for faster execution. Input are projections in npy format for decomposition. Output is a plot with the resulting decomposition.

decofunctions.f95: contains Fortran functions fdot1, mvec, fconv, normalise and nnlsf. fdot1 takes as input two matrices and performs matrix multiplication, mvec multiplies two vectors element wise. fconv takes a vector, a square matrix, the size of the vector and an integer as an input. Output is a square matrix that contains the convolution matrix used in the nnlsf calculation. The convolution matrix can also be set to a -45 degrees projection by shifting all columns around the middle column. This is done if n is set to 1. normalise takes a vector as an input and normalize the vector and removes all elements that are negative. finally, nnlsf is the nonegative square solver implemented in Fortran 90, based on the algorithm proposed by Charles L. Lawson and Richard J. Hanson[23]. Note that the nnls used in numpy is based on the same authors[24] from a revision of the original book.

plotProjections.py: this program reads spectrum and displays their contour levels using matplotlib[25]. input is a complete list of projections and output is a contour map. Apart from the contour map is a cross section of a slice of the spectra. This program is used for visual inspection of the spectra. Contour levels are usually between 0.1 and 1 but can be changed depending on the sensitivity of the spectrum. The spectrum has to be in npy numpy format.

readFT2files.py: a program that reads spectrum in nmrPipe FT2 format and converts them to numpy arrays that are used for decomposition and display of spectra. 512 bytes of header information has to be extracted first.

conversion.py: converts points to ppm for different nuclei

gui3.py: this is the graphical user interface program and it contains several functions. Most functions concerns the graphical functionality and the functions that are used for decomposition are the same as in finalDeco.py

finalDeco.f95: this program was developed to replace the whole decomposition with fortran code. Unfortunately very strange errors made it hard to continue to develop this version. Therefore it is not used and it is not contained in the source appendix.

Evaluation

The code used for the following residues is in the supplementary material. All listed residues reference[26] shifts where compared to the calculated results. The peak picking was made by inspection, i.e to confirm that the shape contained the closest peak to the reference. Normally, final peak picking is done after the sequential assignment is done. In that stage, information from the current and the previous residue are available which creates less ambiguity in the final assignment. The result for the peaks presented above are presented in table 3:

Res.	CO	Cα	Cβ	Hα	Hβ	Cαb	Hαβ
47	176.31	54.84	44.88	4.74	2.66/2.42	57.14/39.66	7.3/3.32/2.39
calc.	176.18	55.94	38.47	4.77	4.18/5.18	55.94/38.47	7.24/3.27/2.36
34	175.17	61.61	69.9	5.08	4.1/3.89	56.59/48.17	5.41/2.0/0.99
calc.	175.14	60.47	34.58	5.03	4.05/2.03	56.59/48.17	5.35/1.96/0.99
103	175.29	58.42	38.59	4.63	1.90/1.78	57.18/46.61	4.51/2.05/1.1
calc.	176.56	58.53	37.17	4.51	2.16/1.90	59.17/32.00	4.44/2.16/--
81	175.62	67.72	42.57	3.71	1.12	66.47/35.52	3.92/2.34
calc.	174.36	66.29	26.17	3.66	2.55	65.00/33.94	3.92/2.29
88	175.93	45.44	--	4.09	--	64.13/46.14	4.89/2.20
calc.	175.92	43.64	--	4.05	--	63.70/--	4.83/--
114	174.01	61.77	43.62	3.47	2.68/2.48	66.3/71.9	4.59/4.95
calc.	173.98	60.47	42.35	3.46	2.62/2.49	--/70.82	--/4.96
38	173.80	57.37	46.46	4.98	2.75/2.6	47.7	5.03
calc.	173.85	55.94	44.94	5.48	2.75/2.55	46.23	5.03
9	175.93	48.46	--	5.19	--	57.52/35.73	5.3/1.73/1.63
calc.	180.06	47.52	--	5.35	--	55.94/63.70	5.22/1.32/1.64
128	175.69	61.65	39.40	4.35	1.73/1.62	57.95/48.24	5.15/1.67/1.34
calc.	175.66	60.47	37.82	4.31	1.71/1.64	57.23/46.88	5.22/1.64/1.32
30	172.78	65.19	75.39	5.11	3.33	60.35/46.89	4.91/2.0/1.61
calc.	171.26	49.47	41.05	5.67	2.35	55.94/--	5.41/3.27/2.23
73	175.53	58.02	50.53	4.57	1.35/1.16	65.04/38.71	3.66/3.09/2.44
calc.	175.40	55.94	48.82	4.31	1.31/12	63.06/34.58	3.98/3.07/2.43
65	178.31	57.59	22.68	4.21	1.57	63.83/36.18	4.04/2.49/2.27

Res.	CO	Cα	Cβ	Hα	Hβ	Cαb	Hαβ
calc.	178.25	55.94	21.00	4.18	1.51	62.41/34.58	3.98/2.36/2.23
18	171.88	56.5	43.72	5.16	3.23/3.17	63.87/73.56	4.78/4.19
calc.	171.78	55.94	47.53	5.16	2.49/3.14	--/72.12	4.70/4.18
84	173.03	65.34	76.00	4.81	5.22	58.32/36.44	6.2/3.44/3.33
calc.	173.33	54.00	21.64	4.05	1.45	54.00/21.64	--/4.05/1.38

Table 3. Results for the algorithm used on the previous residues. Reference ppm are on top and calculated numbers below. Residues 9 and 88 are glycines which lacks Hβ and Cβ are marked with a '--' symbol which also marks missing peaks.

The average of the differences between the calculated values and the reference is presented in table 4. The last row is with the the last residue removed.

CO	Cα	Cβ	Hα	Hβ	Cαβ	Hαβ
0.65	2.98	11.42	0.19	0.62/1.57	1.5/5.09	0.1/0.2/1.19
0.67	2.34	8.11	0.14	0.38/1.4	1.28/4.34	0.11/0.16/0.06

Table 4. Average differences between calculated and reference values with all residues on first row and with the last residue removed on the last row.

The average comparison in table 4 shows that CO had a good average while Cβ was not good. Hα was neither good or bad while Hβ was worse. Overall was Cβ and Hβ worse than Cα and Hα. The reason for this can probably partly be explained by bad experimental conditions. One reason is also the ambiguity for the last residue and the results changed to the better when removing the last residue but not as much as expected. A detailed analysis is done below.

As can be seen in table 3, the result for the 14 selected CO nuclei are in good agreement with the reference values with an average value of 0.66 witch is slightly above a threshold of 0.5 ppm. The exception is residue 30. Residue 30 is close but not whitin a 0.5 ppm threshold and has a peak that is very close and can in fact be the same peak. The CO often gives strong signal in relation to other nuclei mostly because a strong one bond coupling constant thus making it the easiest to calculate of the backbone nuclei. An example of the result of the calculation for residue 9 can be seen in figure 13. The uppermost shape is the resulting CO peak that is selected by multiplying the third shape that is from the single projection on the slice interval x. By multiplying the resulting shape with the projection the N shape can be obtained and this can be used as an aid to see that the calculation is correct.

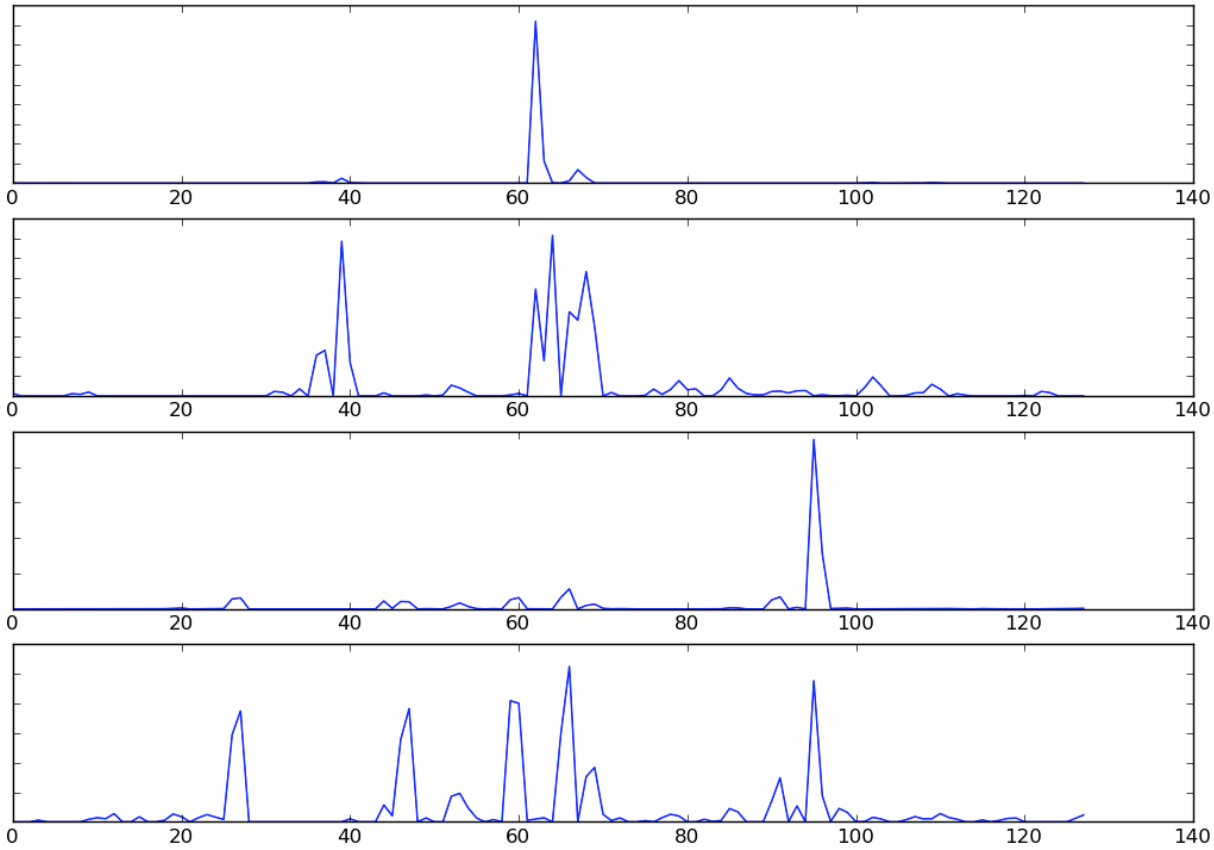


Figure 13. Four shapes that describes the CO and the N nuclei at interval 421. The N shape is at 95 which can be clearly seen in the second panel.

In figure 13 there are at least four shapes representing slices at point 491. The bottom panel show the different N nuclei. The present is at position 95, representing residue 9, a glycine. At least 8 other are present in the panel. The second panel represent the N shape calculated with N+CO and N-CO convolutions using the obtained result of the CO shape from the decomposition. Here it is clear that the right shape indeed is at point 95. The third panel shows the number of possible CO shapes from the convolution calculation. Six peaks are present. The final panel shows the resulting CO shape. This shape is then used to calculate the N shape in order to check that the result is correct.

For the Cα nuclei the result was slightly different. Most residues are out of a 0.5 ppm threshold, a threshold normally used for good comparison. One reason for this is that signal to noise ratio is less than in the CO spectra that have the highest signal to noise ratio. Especially peaks that have a low signal to noise give a lot o signals in the calculation. All these signals can make it hard to select the right peak. The shapes identification for Cα is also complicated by the fact that a Cα peak from the pervious residue is present. This peak is usually with less

intensity but not always. This is because of the type of experiment done and it is also true for Cβ and Hβ. Normally a final peak picking is done when the final backbone sequence is determined, then the Cαβ and Hαβ plus statistical values can be used to increase the shape identification. Another source of error are lack of curve fitting which can shift the ppm slightly.

Residue 46 and 38 are the rightmost and leftmost residues respectively and they should be easy for the algorithm to resolve because they lack neighboring signals that affects the decomposition. However, Cβ and Hβ did not resolve well for residue 46. This was probably because the highest intensity of the Cβ shape was wrong. This can be the fault of measurement errors. The same argument applies the erroneous Hβ. So the algorithm presented the right result but wrong measurement gave errors. Residue 38 gave better result although the N intensity was weaker than residue 46 intensity. Strong intensity in the N shape doesn't have to imply that the rest of the intensity are strong, relaxation effects can reduce the signal for different nuclei whitin a residue. In some places one shape is missing and this can occur when the resolution in the experiment is low and two signals are very close to each other. Significant differences between in table 2 can partly be explained by spectral errors and low signal to noise. One exception is the last residue that have the strongest N intensity of all peaks and still has only CO correct in table 2. This remarkable result is probably because it is wrong reference or it is an artifact. Residue 34 had a clear miss on the Cβ but this peak was actually present with a lower threshold although a lot of other peaks where also present. The same argument is valid for Hβ, here present as a very low intensity peak while the strongest one is clearly wrong. CO for 103 is slightly wrong but the correct peak is present but with a very low intensity. A very weak Cα was present for the Cαβ shape. For residue 81 the Cβ was totally of, here possibly because a strong signal was close to the right signal and this was also true for Hβ. Both Cβ and Hβ where missing in residue 87 possible due to weak signals. Hα on residue 38 was very weak and thus could be considered an artifact. Hβ for the previous residue was very weak compared to the strongest peak but could be peak picked with the help of the previous residue. Glycine residue 9 proved to be very complicated to resolve with a clear miss for the CO shape. A lot of signals where present in the CO plane and the wrong one was picked by the algorithm. Hα was present but with a very low signal making it hard to pick. Also the previous residue gave errors. residue 30 had problems with Cα and Cβ as well as the rest of the residues. Notable is that the comparison score was quite low, 0.21, for this residue. CO for residue 73 was weak but still present. Also the comparison score showed low values, 0.08, indicating wrong shapes. Residue 18 Cβ was weak but present. The previous Hα and Hβ was very weak. The last residue on the list is probably an artifact, the signal i significant stronger than the rest of the signals and it doesn't fit into any residue in the reference list.

Performance

The interfaces used for accessing C and Fortran functions from python where ctype and f2py. cyype uses conversion modules for converting python implicit python types to corresponding c types. The C functions are then called via dynamically linked shared files. Fortran subroutines are called as shared object files. The type conversion between pythons

dynamically typed types and Fortran statically typed types are done via the f2py interface. The Fortran subroutines can then be used as a python module using the import statement. All three languages uses scientific libraries based on lappack. Python uses scipy/numpy, C gnu scientific libraries[27] and Fortran Lapac[28]. Fortran also have inbuilt libraries that handle different matrix manipulations.

The first implementation used ctypes for accessing functions that was slow in the python implementation. These required conversion between the different types to be used in C programs and thus created an unnecessary overhead. Thus this approach was dropped in favor of f2py witch creates shared object files that can be loaded at execution time. Type conversion is minimal and both simple types and combined types as in matrices can be handled easily. Five python functions were replaced with the following Fortran functions: fdot1, mvec, fconv, nnlsf and normalise. Table 5 shows the performance difference between the pure python implementation and the modified implementation using Fortran functions calls. Input was the 71 residues protein ubiquitin[29] and the gui interface was used. First test was on a stationary computer and the second test was on a notebook with a dual core processor 1.8 MHz.

cumulative/s	decompose	decomposeF95
stationary	6.003	4.755
notebook	21.861	5.482

Table 5. Comparison between a pure python implementation and an implementation with Fortran function calls on a stationary computer and a laptop

As can be seen in the table, performance was improved using Fortran functions calls in both computers but the difference is significantly smaller for the stationary computer then the laptop. This means that a computer of todays standard probably would notice small differences on a protein of the size of ubiquitin but with larger proteins the advantage of using Fortran calls would still be present.

Graphical interface

A graphical user interface (gui) was developed to ease the handling of spectra and decomposition. This gui was written in wxPython[30], a graphical library written in python used for developing platform independent graphical interfaces. The gui is used a tool to convert, display, decompose and assign projection experiments. All programs that are part of the gui are listed below. A short description of the program is given here.

1. First the input data from the experiment must be converted and loaded. The supported format are FT files. These files come from the nmrPipe processing package. nmrPipe are used as a processing tool for NMR data and also as a visualizing tool. The corresponded ^{15}N-HSQC has to be peak picked in nmrDraw and a peak list have to be generated that can be used in the gui. Last, an text file generated from the experiment containing parameters

from the experiment have to be loaded and converted. When the peak list is loaded a spreadsheet is generated with the residues and the corresponding nucleus. The resulting shifts will be inserted in this list when the planes are decomposed using the deco algorithm and these shifts can be changed manually as well.

2. The peak list is projected onto the ^{15}N-HSQC plane and both pts and ppm are shown in the lower part of the gui. Also the spectra can be zoomed and the thresholds levels can be changed by using the scroll wheel of the mouse. This spectrum is good to have at hands when analyzing the projections.

3. Decomposition is started either from the menu or from a menu button. The decomposition is done over the whole region defined from the peak list. The algorithm used is described above. The resulting automatically picked peaks are displayed in the spreadsheet when the decomposition is done. When clicking on a cell the resulting decomposition is displayed.

4. Individual residues can be recalculated by first changing their coordinates in the ^{15}N-HSQC spectrum and then clicking on the toolbar and choose single calculation.

5. The final peak list can then be used for a sequence assignment and subsequently a final peak assignment.

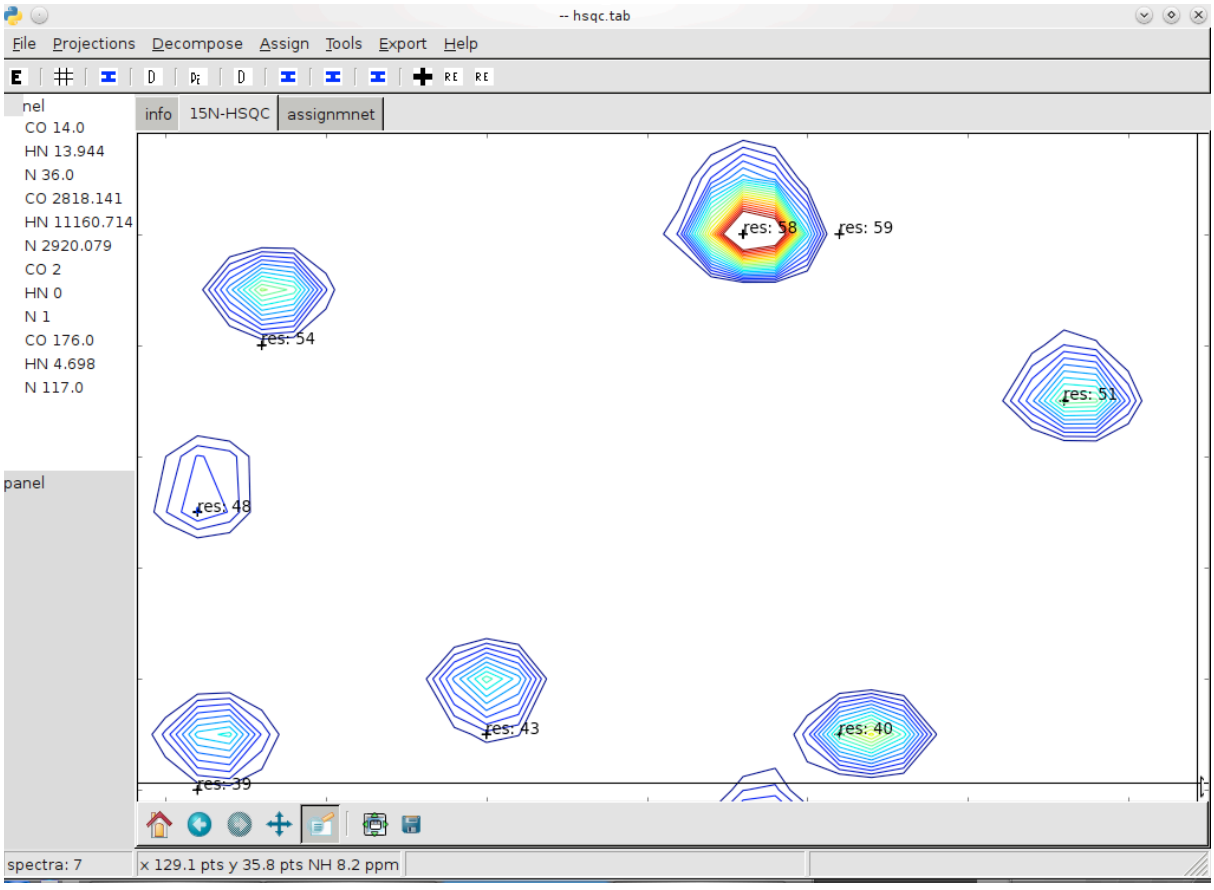


Figure 14. Graphical interface for deco

Conclusions

Deconvolution of projected experiments usually requires several parameters for analysis, interval and number of components are usually required. These parameters make the deconvolution complex and requires a lot of testing before a final interval with a suitable number of components. Also confusing is the fact that components can be used to describe noise or signals. In my approach I consider only one signal at a time and all the rest are considered noise or other neighboring signals. Every interval has a length of five points and this together with the one signal assumption removes the three parameters. The real difficulties in this approach was to separate signals from noise in very dense regions. One of the considerations is that the expected signal can be lost in several projections, thus making it difficult to find the signal. When going from 5D,4D to 3D in effect only four planes can be used, {N,N+X,N-X,X}. The first step here was to use the peak picked information by selecting the picked peak and remove all other signals for N. This approach can compensate for the loss of information to some extent. The inaccuracy in some of the shapes emphasizes the importance of having really good spectra so information is kept as consistent as possible between the experiments.

The final programs used here represent only a subset of the programs used for testing and optimization, several different algorithms where tried to extract the results from the decompositions with varying results. One mistake was to optimize against CO first and then go on to the rest of the nuclei because CO has a much stronger signal intensity and thus gave much better result than the rest of the nuclei. This made the process slower because of the false assumption that the algorithm worked well for the other residues. One other approach was to replace the whole decompose algorithm with a fortran module and this was tested but very strange memory related errors turned up and therefore this was abandoned. The speed gain for this is also questionable. Developing in wxPython was very time efficient and the underlying calls to matplotlib did not create any difficulties. The choice of fortran functions over c functions can be recommended when doing numerical calculations in python because of the easy argument transfer without any explicit type conversion.

Finally I think this algorithm is worthwhile to continue with and that it is suitable for proteins of smaller size with a sufficient large signal to noise.

Acknowledgments

Arne Dalhberg for accepting me working on another continent and for accepting the project

References

[1] K. Wüthrich, Nuclear Magnetic Resonance Spectroscopy of Proteins, Encyclopedia of Life Sciences, (2001) Wiley

[2] G. Petsko, D. Ringe, Protein Structure and Function, (2004) New science Press Ltd

[3] J. H. Prestegard, H. Valafar, J. Glushka, F. Tian, Nuclear Magnetic Resonance in the Era of Structural Genomics, Biochemistry, 40 (2001) 8677-8685

[4] K. Wüthrich, Nuclear Magnetic Resonance Spectroscopy of Proteins, Encyclopedia of Life Sciences, (2001) Wiley

[5] J. Cavanagh, Protein NMR Spectroscopy, second edition. (2007) ELSEVIER

[6] G. Rule, T. Hitchens, Fundamentals of Protein NMR Spectroscopy, (2006) Springer

[7] V. Orekhov, V. Jaravine, Analysis of non-uniformly sampled spectra with Multi-Dimensional Decomposition, Progress in Nuclear Magnetic Resonance Spectroscopy Progr. NMR. Spect. (2011) (doi:10.1016/j.pnmrs.2011.02.002)

[8] M. Billeter, D. K. Staykova, Rapid Multidimensional NMR: Decomposition Methods and their Applications, Encyclopedia of Magnetic Resonance, 9 (2009) Wiley

[9] D. Malmodin, M. Billeter, Robust and versatile interpretation of spectra with coupled evolution periods using multi-way decomposition, Magn. Reson. Chem. 44 (2006) S185–S195

[10] Y. Xia, G. Zhu, S. Veeraraghavan, X. Giao, (3,2)D GFT-NMR experiments for fast data collection from proteins, Journal Of Biomolecular NMR, 29 (2004) 467-476

[11] M.H. Lewitt, Spin Dynamics Basics of Nuclear Magnetic Resonance, (2001) Wiley

[12] D. Malmodin, M. Billeter, Robust and versatile interpretation of spectra with coupled evolution periods using multi-way decomposition, Magn. Reson. Chem. 44 (2006) S185–S195

[13] A.N. Tikhonov, A.A. Samarskij, Equations of mathematical physics, (1990) Dover

[14] Gilat, Amos (2004). MATLAB: An Introduction with Applications 2nd Edition. John Wiley & Sons

[15] Travis E. Oliphant, Python for Scientific Computing, Computing in Science & Engineering 9, (2007), 90

[16] PRODECOMPv3: decompositions of NMR projections for protein backbone and side-chain assignments and structural studies. D.K. Staykova, J. Fredriksson, M. Billeter, Bioinformatics. 24 (2008) 2258-2259

[17] Alex Holkner, ctypes. ctypes run!, The Python Papers, (2007), Volume 2, Issue 2

[18] Anderson GL, Williams J, Hille R, "The purification and characterization of arsenite oxidase from Alcaligenes faecalis, a molybdenum-containing hydroxylase", J. Biol. Chem, (1992), 267 (33): 23674–82. PMID 1331097

[19] Delaglio, F and Grzesiek, S and Vuister, GW and Zhu, G and Pfeifer, J and Bax, A. NMRPipe: A multidimensional spectral processing system based on UNIX pipes. Journal of Biomolecular NMR 6(3):277--293, 1995

[20] F. Lindgren, S. Rännar, Alternative Partial Least-Squares (PLS) Algorithms, Three-Dimensional Quantitative Structure Activity Relationships, (2002), Volume 3, Part I, 105-113, DOI: 10.1007/0-306-46858-1_7

[21] Henk A.L. Kiers, Setting up alternating least squares and iterative majorization algorithms for solving various matrix optimization problems, Computational Statistics & Data Analysis 41 (2002) 157–170

[22] Vladislav Yu. Orekhov a, Victor A. Jaravine b, Analysis of non-uniformly sampled spectra with multi-dimensional decomposition

[23] Charles L. Lawson and Richard J. Hanson, "SOLVING LEAST SQUARES PROBLEMS", Prentice-HalL, 1974

[24] Lawson C., Hanson R.J., (1987) Solving Least Squares Problems, SIAM

[25] John D. Hunter, Matplotlib: A 2D Graphics Environment, Computing in Science & Engineering 9, (2007), 90

[26] J. Leckner, Ph.D. Dissertation, Chalmers University of Technology, (2001) Göteborg, Sweden

[27] M. Galassi et al, GNU Scientific Library Reference Manual (3rd Ed.), ISBN 0954612078

[28] E. Anderson et al, LAPACK Users' Guide Third Edition (1999) SIAM

[29] A. Hershko, A. Ciechanover, The Ubiquitin System, Annu. Rev. Biochem. 67 (1998) 425–79

[30] Hugues Talbot, wxPython, a GUI Toolkit, Linux Journal, Volume 2000 Issue 74es, June 2000